5

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/992,130	11/14/2001	Ronald Hilton	AMDH-08152US0 DEL	4626

21603 7590 05/21/2007
DAVID E. LOVEJOY, REG. NO. 22,748
102 REED RANCH ROAD
TIBURON, CA 94920-2025

EXAMINER	
SAXENA, AKASH	
ART UNIT	**PAPER NUMBER**
2128	

MAIL DATE	DELIVERY MODE
05/21/2007	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)
	09/992,130	HILTON, RONALD
	Examiner	Art Unit
	Akash Saxena	2128

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *11 March 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-3,5-17 and 19-28* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-3,5-17 and 19-28* is/are rejected.

7)☒ Claim(s) *10,24 and 26* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. Claim(s) 1-3, 5-17 and 19-28 has/have been presented for examination based on amendment filed on 11th March 2007.
2. Claim(s) 1, 11, 15 and 25 is/are amended.
3. Claims 10, 24 and 26 remain objected to at present.
4. Examiner withdraws the claim rejection(s) under 35 USC § 112 to claim(s) 1, 11, 15 and 25 in view of applicant's amendment to claims.
5. Claim(s) 1-3, 5-17 and 19-28 are rejected under 35 USC § 103 under new grounds of rejection in view of claim amendment with Mann'295, Smith'1982, Wing and Scalzi'013.
6. The arguments submitted by the applicant have been fully considered. Claims 1-3, 5-17 and 19-28 remain rejected and this action is made FINAL. The examiner's response is as follows.

Response to Applicant's Remarks for 35 U.S.C. § 112/103

7. Response to Section 10

No argument presented other than stating the examiner's previous rejection and alleging that TLB cannot perform the function of indexing table. Examiner maintains that TLB and indexing table are functionally similar and no distinction is presented in the disclosure. The figure presented to show the distinction between TLB and Indexing table on Pg. 17 of current remarks are not part of the disclosure. Secondly, these figures only buttress applicant's written arguments cyclically and otherwise are not presented with a supporting prior art documentation showing this distinction is well known in art.. Examiner does not consider these arguments persuasive and maintains the rejection.

8. Response to Section 11.1 –11.3

Applicant is arguing that examiner has shifted position (as in Section 11.1-11.2), and there is a significant (as in Section 11.3) distinction between the examiner's arguments presented on 5/18/06 and 1/12/07.

Examiner respectfully disagrees that position is shifted. The position presented in the office action dated 5/18/06 for Mann'295 was that it does not explicitly teach many-to-one mapping of the legacy instruction 70 to the host code 88 (See Mann'295: Fig.3). Upon further review of the Figure 3 in Mann'295, a detailed understanding of the how the many-to-one translation happens using the block tables 80 shows that Mann'295 is aware of such a translation but does not show the exact details for implementation of the such a block entry table, now mapped as TLB in the Smith'1982 curing Mann'295's deficiency.

9. Response to Section 11.4

As per applicant's arguments to one-to-one mapping an details of block table (as in Section 11.4), they are presented to clarify the point made above. Applicant seems to be ignoring the analysis that shows block entry table show many-to-one mapping between the block table and host code, where block entry table are associated on one to one basis with the legacy instructions, hence meeting the limitation of legacy instructions having one-to-many relationship with host code.

To restate:

Legacy Instruction ←one-to-one-> block entry table <-many-to-one-> host code.

10. Response to Section 11.5

Applicant states that the mapping is one to one, simple and there is no problem to be solved. Examiner respectfully disagrees.

What is clear from Mann'295 Fig.3:

(a) In Fig.3, Table 80 is the translation-mapping table between the legacy code 70 and host code 86/88.

(b) First table 80 and second table 80, translate different segments of legacy code.

(c) Cross-referencing links between first Table 80 and second Table indicate that there is a mapping in the first Table 80 that need to jump to second Table 80 and vise versa.

(d) The first Table 80 and the second Table 80 do not execute from the same mapped code – i.e. start code execution from different points in the host code.

<u>Now the problem to be solved based on what is already established in (a) to (d):</u>

(a) and (b) lay the grounds/structure for the translation table 80. The basis (c) clearly shows that reuse of the host code, because jumping from first to second table 80 is futile without executing associated host code (supported by basis (d)). The basis (b) also shows that instructions 76 are associated with different tables, without specifying if the instructions are identical (at least in figure.3), here is the obvious assertion (i): that executing same instruction (i.e. specifically same instruction with same operand and same data) is futile. Hence to get any meaningful translations, instructions have to have different data at least. Here is the where the implementation details are missing in Mann'295. As stated in previous office action:

> To reassert, the important point is that host code 88 is reused by many block table entries 70 (instructions). In any practical implementation of instruction reuse same data would not be executed (i.e. instruction reuse does not imply data executed is also reused, otherwise the exercise may be futile – running same instruction with same data again and again). This is also precisely the problem not addressed by Mann'295 where there is no distinction made between the instruction and the data in the host code 88, at least in the disclosure.
>
> To make and use the teaching of Mann'295, a Table 80 has to be implemented. TLB (Translation Lookaside Buffer) are practical implementation of the Table 80 which maps the virtual address of the block table entries 70 to real address in host code 88. Smith'1982 discloses the problem of "synonym" as multiple block table entries 70 mapping to same host code 88 (Smith'1982: Pg. 510 section 2.9).

Mann'295 does not disclose implementation detail, hence is simplistic, and requires Smith'1982 to cure its deficiencies.

Further in response to applicant's argument that the Mann'295 does not have a problem. Examiner recites the arguments presented in previous office action:

> Applicant argues that examiner finds the problem being solved in Mann'295 which is not based on the operations presented in Mann'295. Any practical disclosure or teaching, does not discredit itself, in fact it shows the deficiencies in the prior art. Hence the deficiency in Mann'295 would not be apparent from Mann'295. Mann'295 discloses and claims a "methodology" for code emulation, not the exact details of the implementation (e.g. implementation of Table 80). If one would implement the methodology, for example a Table 80 in Fig. 3, an exemplary practical implementation of such a table through TLB, it would bring in implementation issues, which are not limited by which table structure is used (TLB, indexing table etc). Hence to make and use the teaching of Mann'295 considerations presented in Smith'1982 must be taken into account.
>
> Applicant alleges that the reason Mann'295 would not face the issues disclosed in Smith'1982 is operation of Mann'295 are not as suggested by examiner. Examiner asserts that such a manner is not only asserted by examiner, but it is also disclosed by Mann'295 as explained above. Hence combination of Mann'295 with Smith'1982 would be obvious to one skilled in the art of code emulation.

Examiner finds applicant's arguments unpersuasive.

11. Response to Section 11.6-11.8

Applicant has stated:

> Nothing in Mann '295 describes an operation wherein for some first target instruction and some second target instruction (either from a single target block or from multiple target blocks), the first

and second target instructions are mapped to first host instructions and second host instructions in a host code block wherein the <u>host code address spaces for the first host instructions potentially overlap or otherwise conflict with the host code address spaces for the second host instructions</u>. Rather, Mann '295 merely describes <u>(1)</u> a one-to-one single target block to single host block structure and <u>(2)</u> a one-to-one single target instruction to a single set of one or more host code instructions for emulating the single target instruction. Nothing in Mann '295 suggests any operation wherein a host code address conflict might arise from a many-to-one mapping.

Applicant argument regarding Mann'295 teaching the overlap in the host address space for the first set of instructions and second set of instructions further buttresses examiner's assertion that <u>*many*</u> legacy instructions reuse the <u>*same*</u> (one) host code (therefore a many to one mapping).

The statements (1) & (2) above, directly contradicts the underlined statement preceding it.

Further in the last statement applicant states:

Nothing in Mann '295 suggests any operation wherein a host code address conflict might arise from a many-to-one mapping.

As clear from the analysis there <u>*is*</u> many-to-one mapping between the legacy instructions to host instruction, applicant acknowledges that there may be host code address conflict. This is the basis as to why Mann'295 is deficient. Mann'295 teaching when implemented would need to be supplemented by Smith'1982 as stated in the Response to Section 11.5 above.

As for 11.7 & 11.8, use of problem in Mann'295 and Smith'1982 is justified above using applicant's own understanding of Mann'295 reference.

12. <u>Response to Section 12</u>

The examiner notes applicant's comments.

13. Response to Section 13

No comment

14. Response to Section 14.1-14.3

Firstly, examiner has previously explained many-to-one mapping in response to sections 11.4, 11.6-11.8 above. Secondly, applicant's assertion and claim, that "storing translation indications using a subset of block address digits whereby block numbers in said table are same for multiple different blocks", does not correspond with disclosure (See Fig.4 TC Table looking at column BN where no block numbers are repeated). Therefore examiner maintains that Mann'295 clearly teaches the limitation as recited in the rejection.

15. Response to Section 14.4

As to comment on interpretation, and amended claim please see updated claim rejection.

16. Response to Sections 15-18

Arguments are noted. No comments. Please see updated rejection.

17. Response to Sections 19

Please see updated claims relating to comment on interpretation.

18. Responses to Section 20

Examiner maintains the position as presented in 12.7.4.1, 12.11, 13, 14 and 16.

19. Response to Section 21

Examiner withdraws the rejection under 35 USC 112-second paragraph, in view of claim amendment. The remark about join inventorship is noted and corrected.

20. Response to Section 22.1-22.4.1

As to comment on interpretation, and amended claim please see updated claim rejection.

21. Response to Section 22.5

Applicant has stated that A and B steps are not taught by Mann'295 and Smith'1982. Examiner respectfully disagrees, and maintains the rejection as previously presented. Arguendo, even if that is true the updated rejection with Wing teaches tag T that updates the self modifying code leads to execution of the translated legacy instruction if not modified and retranslation (T tag getting marked as stale) if the self modifying code is encountered (See Wing Col.22 Lines 18-43).

22. Response to Section 25-26

Applicant's remarks are noted regarding allowable subject matter. Regarding deficiency in Mann'295, updated rejection is presented in view of amended claim and explanation as presented in Sections 22.1 to 22.4.1.

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

The factual inquiries set forth in *Graham* **v.** *John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966), that are applied for establishing a background for determining obviousness under 35 U.S.C. 103(a) are summarized as follows:

1. Determining the scope and contents of the prior art.
2. Ascertaining the differences between the prior art and the claims at issue.
3. Resolving the level of ordinary skill in the pertinent art.
4. Considering objective evidence present in the application indicating obviousness or nonobviousness.

23. **Claims 1-3, 5, 7-9, 11-13, 15-17, 19, 21-23, 25, and 27 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6516295 issued to George A. Mann et al (Mann '295 hereafter), further in view of ACM Article "Cache Memories" by Alan Jay Smith (Smith '1982 hereafter), further in view of U.S. Patent No. 5,926,832 by Wing et al (Wing hereafter).**

Regarding Claim 1 (Updated 5/15/07)

Mann '295 teaches a *computer-implemented* method of emulating execution of a legacy *instruction including self-modifying code* (Mann '295: Col.2 Lines 44-51). Mann '295 teaches instructions having instruction address (Mann '295: Col.5 Lines 28-29).

Further, **Mann '295** teaches accessing blocks of legacy instruction (Mann '295: Col.6 Lines 11-12). **Mann '295** teaches blocks having block addresses (Mann '295: Col.6 Lines 17-19).

Further, **Mann '295** teaches storing *translations* into translation store as the host code block (Mann '295: Fig. 3 Element 88; Col.5 Lines 58-63; Col.6 Lines 11-28) for each legacy instruction.

Further, **Mann '295** teaches storing translation indication at block numbers determined by block addresses (Mann '295: Fig. 3 Element 81). Mann '295 teaches indexing table as block entry table for indicating that the block has been translated (Mann '295: Col.6 Lines 62-66).

Further, **Mann '295** teaches each particular legacy instruction of the translated block having a particular block number (Mann '295: Fig. 3 Element 72F-L, 80-81).

Further, **Mann '295** teaches translating a particular legacy instruction into one or more translated instructions for emulating the particular legacy instruction (Mann '295: Col.6 Lines 53-55).

Further, **Mann '295** teaches if a legacy instruction is not a store instruction, going to step of executing translated instruction as performing the DOCT (Mann '295: Col.9 Lines 5-10; Fig. 4 Elements: 126, 124, Fig. 5 Element 134/136 & 148).

Mann '295 teaches if the instruction is a store instruction, where the store instruction stores to a particular legacy block with a particular block number in the (block translation) table (Mann '295: Fig.4; Col.9 Lines 10-36, Col.7 Line 49-Col.8 Line 65).

~~**Mann '295** teaches if the indication indicates that said particular block (of legacy instruction) has not been translated (indicated as X – do not translate), going to the step of executing the translated instruction (Mann '295: Fig.4 Path 108, 128, 129).~~

Mann '295 teaches if the indication indicates that said particular block (of legacy instruction) has not been translated <u>to cause code modification</u>, going directly to the step of executing the translated instruction as Mann'295 Col.7 Lines 4-13

> Other special handling is also typically required, especially if the modified instruction is in the middle of a Target block of code. <u>Under normal circumstances, when program execution is not currently in the Host code block 88 corresponding to the Target code block containing the modified instruction, it is sufficient to mark the entire code block with "X" and deallocate the corresponding Host code block 88</u>. <u>Alternatively, the Host code block 88 can be split into two Host code blocks 88: one before the modified code, and one after</u>.

Man'295 teaches when there is <u>cause for code modification</u> there are two ways for handling the code, otherwise the execution of the translated code will continue as shown (when code is not marked with X, i.e. Tag=F/E scenario, Fig.1 102, 110, 120, 124).

<u>*On a closer review*</u>, Mann'295 also teaches using index/offset <u>(partial addresses)</u> from the instructions as index for the block entry table, thereby teaching the limitation of "said storing translation indications using a subset of block address digits whereby block numbers in said table are the same for multiple different blocks" (Mann'295: Col. 6 Lines 16-24).

Mann'295 <u>also does not teach</u> the details of the limitation presented above explicitly.

Smith '1982 teaches that TLB having a hashing mechanism to map the virtual addresses (block numbers) to the real address (translated host code block) (Smith '1982: Pg.475 Col.2 Paragraphs 3-4). Hashing (done by taking an XOR or through randomized algorithm) depends on the number of bits selected, resulting in folding or overlapping (Smith '1982: Pg.488, Col.1 Paragraph 1; Pg.489, Col.1). The hashing scheme selected by the applicant is extremely simplified version, as only one middle bit is selected to index the translation indication index table.

***Mann '295** teaches checking if the instruction has been translated so as to possibly cause code modification (Mann '295: Table T1, Non-X statuses; Fig.4), checking translation store to determine if the ~~legacy instruction data~~ code has been modified, <u>if the code has been modified</u>, repeating the translation of legacy instruction (in new host code block starting with code 'F') then executing the instruction (Mann '295: Col.7 Lines 4-38; Fig.4; Col.9 Lines 9 –20). If the instruction data has not been modified then executing the translated instructions (Mann '295: Fig.4 Path 102, 110, 120, 124).*

Mann'295 *<u>does not teach repeating the step of translation (re-translation) of the self-modifying code.</u> It marks it with tag "X" and executes interpretation.*

Wing *teaches repeating the step of translation (re-translation) of the self-modifying code (See Wing: Col.6 Lines 42-51- Problem description; Col.22 Lines 18-43 solution by marking the data in the TLB as stale translated code and raising exception; Col.10 Line 65-Col.11 Line 6 – handling exceptions by retranslating the code).* **Wing** *also* teaches a <u>*computer-implemented*</u> method of emulating execution of a legacy <u>*instruction including self-modifying code*</u> (Wing: *Col.22 Lines 18-43*).

Mann '295 *teaches executing translated instructions to emulate the legacy instruction (Mann '295: Fig.4 Element 124).*

It would have been obvious to one (e.g. a designer) of ordinary skill in the art at the time the invention was made to combine the teachings of <u>Smith '1982 and apply them to Mann '295</u> to implement the indexing table as disclosed. The motivation would have been that Smith '1982 discloses the necessity for TLB like lookup when dealing with translated information when address space doesn't map directly (Smith '1982: Pg.510, Section 2.9, 2.17). Mann '295's design requires translation as one target code block may have one or more translated host code instructions. Hence Smith '1982 solves Mann '295's problem of mapping the target legacy instruction object to translated host object code block (Mann '295: Col.6 Lines 47-55). Please see further the response to arguments for clarification.

It would have been obvious to one (e.g. a designer) of ordinary skill in the art at the time the invention was made to combine the teachings of <u>Wing and apply them</u>

*to Mann '295 to handle self modifying code through translation rather than interpretation. The motivation to combine would be that such translation is much faster and in performance using code morphing software and uses a similar mapping TLB structure as **Mann'295** (See Mann'295: Fig.4; Wing: Col.20 Lines 29-37; Col.22 Lines 18-43). Further both Mann'295 and Wing are analogous art in field of code emulation using translation with ability to handle self modifying code, although in different manners.*

Regarding Claim 2

Mann '295 teaches the step of storing translation indications for only a subset of all translated blocks (Mann '295: Col.5 Lines 54-63).

Regarding Claim 3

Mann '295 storing the translated executable host code on the volatile cache memory like SRAM (Mann '295: Col.3 Lines 36-38, 55-61; Col.4 Lines 16-18).

Regarding Claim 5

From the teachings of Mann'295 and Smith'1982 it obvious to use the subset (middle 3 hexadecimal) of digits form the legacy instruction for the part of the address field of the indexing table (Mann'295: Col. 6 Lines 16-24; Smith'1982: Pg.475 Col.2 Paragraphs 3-4; Pg.488, Col.1 Paragraph 1; Pg.489, Col.1).

Regarding Claim 7

Mann '295 teaches that legacy instructions are object code instructions compiled/assembled for the legacy architecture (Mann '295: Col.2 Lines 44-51).

Regarding Claim 8

Mann '295 teaches legacy instruction includes store instructions for modifying instruction code (Mann '295: Col.9 Lines 5-20, 38-47; Col.7 Lines 14-37).

Regarding Claim 9

Mann '295 teaches translation indication includes a state field as tag field, for each block number, indicating the block has been modified (Mann '295: Col.5 Table 1; Col.6 Lines 62-67).

Regarding Claim 11

Mann '295 teaches a method of dynamic emulating execution of a legacy instruction (Mann '295: Col.2 Lines 44-51). Mann '295 teaches instructions having instruction address (Mann '295: Col.5 Lines 28-29). Further, **Mann '295** teaches accessing blocks of legacy instruction (Mann '295: Col.6 Lines 11-12). **Mann '295** teaches blocks having block addresses (Mann '295: Col.6 Lines 17-19). Further, **Mann '295** teaches storing *translations* into translated code into host code block (translation store) for each legacy instruction (Mann '295: Fig. 3 Element 88; Col.5 Lines 58-63; Col.6 Lines 11-28).

Further, **Mann '295** teaches storing translation indication at block numbers determined by block addresses (Mann '295: Fig. 3 Element 81). Mann '295 teaches indexing table as block entry table for indicating that the block has been translated (Mann '295: Col.6 Lines 62-66).

Mann '295 teaches executing translated instructions to emulate the legacy instruction (Mann '295: Fig.4 Element 124).

Further, **Mann '295** teaches each particular legacy instruction of the translated block having a particular block number (Mann '295: Fig. 3 Element 72F-L, 80-81).

Further, **Mann '295** teaches translating a particular legacy instruction into one or more translated instructions for emulating the particular legacy instruction (Mann '295: Col.6 Lines 53-55).

Further, Mann '295 teaches checking store instruction associated to the block entry table, if instruction data is stored (Mann '295: Col.9 Lines 5-10; Fig. 4 Elements: 126, 124, Fig. 5 Element 134/136 & 148) for a particular block. Mann '295 also teaches checking if the instruction data has been modified (Mann '295: Col.9 Lines 9 –20). Further, Mann '295 teaches bypassing the checking if there is no store instruction data (Mann '295: Fig. 5 Element 134/136 & 148). Further, **Mann '295** teaches the step of storing translation indications for only a subset of all translated blocks (Mann '295: Col.5 Lines 54-63).

Mann '295 teaches if the instruction is a store instruction, where the store instruction stores to a particular legacy block with a particular block number in the (block translation) table (Mann '295: Fig.4; Col.9 Lines 10-36, Col.7 Line 49-Col.8 Line 65).

~~**Mann '295** teaches if the indication indicates that said particular block (of legacy instruction) has not been translated (indicated as X – do not translate), going to the step of executing the translated instruction (Mann '295: Fig.4 Path 108, 128, 129).~~

Mann '295 teaches if the indication indicates that said particular block (of legacy instruction) has not been translated <u>to cause code modification</u>, going directly to the step of executing the translated instruction as Mann'295 Col.7 Lines 4-13

> Other special handling is also typically required, especially if the modified instruction is in the middle of a Target block of code. Under normal circumstances, when program execution is not currently in the Host code block 88 corresponding to the Target code block containing the modified instruction, it is sufficient to mark the entire code block with "X" and deallocate the corresponding Host code block 88. Alternatively, the Host code block 88 can be split into two Host code blocks 88: one before the modified code, and one after.

Man'295 teaches when there is cause for code modification there are two ways for handling the code, otherwise the execution of the translated code will continue as shown (when code is not marked with X, i.e. Tag=F/E scenario, Fig.1 102, 110, 120, 124).

Mann '295 *teaches checking if the instruction has been translated so as to possibly cause code modification (Mann '295: Table T1, Non-X statuses; Fig.4), checking translation store to determine if the ~~legacy instruction data~~ code has been modified, if the code has been modified, repeating the translation of legacy instruction (in new host code block starting with code 'F') then executing the instruction (Mann '295: Col.7 Lines 4-38; Fig.4; Col.9 Lines 9 –20). If the instruction data has not been modified then executing the translated instructions (Mann '295: Fig.4 Path 102, 110, 120, 124).*

Mann'295 *does not teach repeating the step of translation (re-translation) of the self-modifying code. It marks it with tag "X" and executes interpretation.*

Wing *teaches repeating the step of translation (re-translation) of the self-modifying code (See Wing: Col.6 Lines 42-51- Problem description; Col.22 Lines 18-43 solution by marking the data in the TLB as stale translated code and raising exception; Col.10 Line 65-Col.11 Line 6 – handling exceptions by retranslating the* *code).* **Wing** *also* teaches a _computer-implemented_ method of emulating execution of a legacy _instruction including self-modifying code_ (Wing: *Col.22 Lines 18-43*).

***Mann '295** teaches executing translated instructions to emulate the legacy instruction (Mann '295: Fig.4 Element 124).*

Mann '295 teaches storing the translated executable host code on the volatile cache memory like SRAM (Mann '295: Col.3 Lines 36-38, 55-61; Col.4 Lines 16-18).

Mann '295 teaches translation indication includes a state field as tag field for each block number indicating the block has been modified (Mann '295: Col.5 Table 1; Col.6 Lines 62-67).

Regarding Claim 12

Method claim 12 is directed towards similar limitations as the method claim 10 and is rejected for the same reason as claim 10.

Regarding Claim 13

Mann '295 teaches that legacy instructions are object code instructions compiled/assembled for the native legacy architecture executed as guest on host architecture (Mann '295: Col.2 Lines 44-51, Fig.3).

Regarding Claim 15 (Updated as in Claim 1)

Mann '295 teaches an apparatus and a method for emulating self-modifying code. The system claim 15 is directed towards the same limitations as the method claim 1 and is rejected for the same reason as claim 1. ***Wing** teaches repeating the step of translation (re-translation) of the self-modifying code (See Wing: Col.6 Lines 42-51- Problem description; Col.22 Lines 18-43 solution by marking the data in the TLB* *as stale translated code and raising exception; Col.10 Line 65-Col.11 Line 6 – handling exceptions by retranslating the code).* **Wing** *also* teaches a *computer-implemented* method of emulating execution of a legacy *instruction including self-modifying code* (Wing: *Col.22 Lines 18-43*).

Regarding Claim 16

The system claim 16 is directed towards the same limitations as the method claim 2 and is rejected for the same reason as claim 2.

Regarding Claim 17

The system claim 17 is directed towards the same limitations as the method claim 3 and is rejected for the same reason as claim 3.

Regarding Claim 19

The system claim 19 is directed towards the same limitations as the method claim 5 and is rejected for the same reason as claim 5.

Regarding Claim 21

The system claim 21 is directed towards the same limitations as the method claim 7 and is rejected for the same reason as claim 7.

Regarding Claim 22

The system claim 22 is directed towards the same limitations as the method claim 8 and is rejected for the same reason as claim 8.

Regarding Claim 23

The system claim 23 is directed towards the same limitations as the method claim 9 and is rejected for the same reason as claim 9.

Regarding Claim 25 (Updated as in claim 1)

The system claim 25 is directed towards the same limitations as the method claim 11 and is rejected for the same reason as claim 11. Further, Mann '295 storing the translated executable host code on the volatile cache memory like SRAM (Mann '295: Col.3 Lines 36-38, 55-61; Col.4 Lines 16-18). ***Wing*** *teaches repeating the step of translation (re-translation) of the self-modifying code (See Wing: Col.6 Lines 42-51- Problem description; Col.22 Lines 18-43 solution by marking the data in the TLB as stale translated code and raising exception; Col.10 Line 65-Col.11 Line 6 – handling exceptions by retranslating the code).* **Wing** *also* teaches a *computer-implemented* method of emulating execution of a legacy *instruction including self-modifying code* (Wing: *Col.22 Lines 18-43*).

Regarding Claim 27

The system claim 27 is directed towards the same limitations as the method claim 13 and is rejected for the same reason as claim 13.

24. Claims 6, 14, 20 and 28 are rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6516295 issued to George A. Mann et al (Mann '295 hereafter), further in view of ACM Article "Cache Memories" by Alan Jay Smith (Smith '1982 hereafter), further in view of U.S. Patent No. 5,926,832 by Wing et al (Wing hereafter), further in view of U.S. Patent No. 5,560,013 issued to Casper A. Scalzi et al (Scalzi '013 hereafter).

Regarding Claim 6

Teachings of Mann '295, Smith'1982 and Wing are disclosed in the claim 1 rejection above.

Mann '295, Smith'1982 and Wing do not explicitly teach legacy instructions are for a legacy system having S/390 Architecture.

Scalzi '013 teaches that legacy instructions are for a legacy system having S/390 Architecture (Scalzi '013: Col.17 Lines 54-57).

Motivation to combine Mann'295 with Smith'1982 is same as in claim 1.

Motivation to combine Mann'295 with Wing is same as in claim 1.

It would have been obvious to one (e.g. a designer) of ordinary skill in the art at the time the invention was made to combine the teachings of Scalzi '013 and apply them to Mann '295 to emulate execution of legacy instruction for S/390 legacy architecture. The motivation would have been that Scalzi '013 and Mann '295 are analogous art and Scalzi '013 is performing the instruction set translation in a very similar fashion as Mann '295 through mapping/dynamic address translation (Scalzi

'013: Col. 5 Lines 17-23) and instruction-self-modification (Scalzi '013: Col.12 Lines11-24; Col.14 Lines 16-24).

Regarding Claim 14

Teachings of Mann '295 & Smith '1982 are disclosed in the claim 11 rejection above.

Mann '295 & Smith '1982 do not teach emulated execution by translation from CISC based legacy instruction set to the RISC based target/host instruction set.

Scalzi '013 teaches that legacy instructions are for a legacy system having S/390 Architecture which is a CISC architecture and host architecture is power PC (RISC) based architecture (Scalzi '013: Col.17 Lines 54-57).

It would have been obvious to one (e.g. a designer) of ordinary skill in the art at the time the invention was made to combine the teachings of Scalzi '013 and apply them to Mann '295 to emulate execution of legacy instruction for S/390 legacy architecture. The motivation would have been that Scalzi '013 and Mann '295 are analogous art and Scalzi '013 is performing the instruction set translation in a very similar fashion as Mann '295 through mapping/dynamic address translation (Scalzi '013: Col. 5 Lines 17-23) and instruction-self-modification (Scalzi '013: Col.12 Lines11-24; Col.14 Lines 16-24).

Regarding Claim 20

The system claim 20 is directed towards the same limitations as the method claim 6 and is rejected for the same reason as claim 6.

Regarding Claim 28

Teachings of Mann '295 & Smith '1982 are disclosed in the claim 25 rejection above. The system claim 28 is directed towards the same limitations as the method claim 14 and is rejected for the same reason as claim 14. To facilitate prosecution rejection is repeated below.

Mann '295 & Smith '1982 do not teach emulated execution by translation from CISC based legacy instruction set to the RISC based target/host instruction set.

Scalzi '013 teaches that legacy instructions are for a legacy system having S/390 Architecture which is a CISC architecture and host architecture is power PC, RISC based architecture (Scalzi '013: Col.17 Lines 54-57).

It would have been obvious to one (e.g. a designer) of ordinary skill in the art at the time the invention was made to combine the teachings of Scalzi '013 and apply them to Mann '295 to emulate execution of legacy instruction for S/390 legacy architecture. The motivation would have been that Scalzi '013 and Mann '295 are analogous art and Scalzi '013 is performing the instruction set translation in a very similar fashion as Mann '295 through mapping/dynamic address translation (Scalzi '013: Col. 5 Lines 17-23) and instruction-self-modification (Scalzi '013: Col.12 Lines11-24; Col.14 Lines 16-24).

Allowable Subject Matter

25. Claims 10, 24 and 26 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

The following is a statement of reasons for the indication of allowable subject matter: Mann '295 does not teaches incrementing/decrementing the count for the modified/removed block, but teaches equivalent functionality of removing and de-allocating the block entry table (indexing table entry) when the translated host block is no longer needed/garbage collected (Mann '295: Col.7 Lines 62-67, Col.8 Lines 1-3). Mann '295 also teaches that the translation indication includes a state field (tag field), for each block number, indicating the block has been modified (Mann '295: Col.5 Table 1; Col.6 Lines 62-67). Further, Mann '295 teaches incrementing the state field each time block is executed on the cache (Mann '295: Col.6 Lines 64-65). Therefore the count is not associated to the addition or removal of block.

Conclusion

26. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Communication

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Akash Saxena whose telephone number is (571) 272-8351. The examiner can normally be reached on 9:30 - 6:00 PM M-F.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kamini S. Shah can be reached on (571)272-2279. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Akash Saxena
Patent Examiner, GAU 2128
(571) 272-8351
Tuesday, May 15, 2007

Fred Ferris
Primary Examiner, GAU 2128
Structural Design, Modeling, Simulation and Emulation
(571) 272-3778

[illegible]

FRED FERRIS
PRIMARY EXAMINER
TECHNOLOGY CENTER 2100